TD 2 : Machine de Peano

1 Définition : le modèle de base

La machine de Peano est un évaluateur d'expressions. Avec le modèle de base, les seules expressions disponibles sont :

- **0** : son évaluation donne ... 0 ;
- $++\,\mathbf{E}$ (où E est une expression) : son évaluation donne le résultat de l'évaluation de E, plus 1 ;
- $--\,\mathbf{E}$ (où E est une expression) : son évaluation donne le résultat de l'évaluation de E, moins 1. Si le résultat de l'évaluation de E est 0, $--\,\mathbf{E}$ a une valeur indéterminée ;
- **appel de fonction** : on peut définir de nouvelles fonctions, et les appeler pour s'en servir. Si f est une fonction que l'on a déjà définie, prenant n arguments (éventuellement 0), un appel de f est de la forme $f(E_1, \ldots, E_n)$ où $E_1, \ldots, E_n$ sont des expressions. L'évaluation de l'appel est le résultat de l'application de f aux résultats des évaluations de $E_1, \ldots, E_n$.

Définition de fonction : une fonction est définie par

$$f(X_1, \ldots, X_n) = \textbf{Définition}$$

où **Définition** est

- soit une expression,
- soit un des arguments d'appel (l'un des X_i),
- soit un *Si... Alors... Sinon* :
  si Expression1 = Expression2 alors Définition sinon Définition

Exemple : la fonction `plus` peut être définie par

```
plus(X,Y) = si X = 0 alors Y sinon ++plus(--X, Y)
```

1.1 Les entiers (le début au moins)

Écrire les fonctions (sans argument) `UN`, `DEUX`, `TROIS` dont les résultats sont respectivement 1, 2, 3.

1.2 Un autre algorithme pour l'addition

Un autre algorithme pour additionner `X` et `Y` est basé sur l'idée qu'il suffit d'incrémente `Y` `X` fois. Pour cela, on a besoin d'un compteur `Nb_incr` qui indique le nombre de fois que `Y` a été incrémenté. Quand ce compteur atteint `X`, le résultat de l'addition est dans `Y`. On peut donc écrire

```
plus(X,Y) = plus_aux(X,Y,0)
```

où `plus_aux` est la fonction qui implémente l'algorithme décrit. Écrire cette fonction `plus_aux(X,Y,Nb_incr)`

1.3 Soustraction

Écrire la fonction `moins(X,Y)` qui renvoie $X - Y$. Si $X < Y$ le résultat est indéterminé.

1.4 Décrémentation

La fonction `plus_aux` de la question 1.2 peut être utilisée avec des valeurs quelconques pour ses arguments. Que renvoie l'appel `plus_aux(X,0,Y)` ? Exprimer, en fonction de `X`, `Y` et `Z`, le résultat de l'appel `plus_aux(X,Y,Z)`.

En déduire la fonction `moins_un(X)` qui renvoie la valeur de X, moins 1, si $X > 0$, et dont le résultat est indéterminé si $X = 0$.

1.5 Comparaison

Écrire la fonction `inférieur(X,Y)` qui renvoie 1 si $X \leq Y$ et 0 sinon.

2 Modèle LUXE

Pour à peine plus cher que le modèle de base, on peut s'offrir le modèle LUXE de la Machine de Peano. Ce modèle comprend, en plus des instructions de la machine de base,

- les opérateurs arithmétiques + (addition), − (soustraction), × (multiplication), / (division entière) et % (reste) ;
- les comparaisons $Exp_1 < Exp_2$, $Exp_1 \leq Exp_2$, $Exp_1 > Exp_2$, $Exp_1 \geq Exp_2$, $Exp_1 \neq Exp_2$.

2.1 Primalité

Écrire la fonction `sans_diviseurs_entre(G,D,N)` qui renvoie 1 si N n'a aucun diviseur à la fois plus grand que G et plus petit que D, et 0 sinon. En déduire la fonction `premier(N)` qui renvoie 1 si N est premier et 0 sinon.

2.2 PGCD

Écrire la fonction `euclide(N,M)` qui renvoie le plus grand commun diviseur de N et M, en utilisant l'algorithme d'Euclide.

3 Modèle GRAND LUXE

Au sommet de la gamme des machines de Peano, le modèle GRAND LUXE offre toutes les possibilités du modèle LUXE, auquel on ajoute des opérations sur les listes d'entiers :

- l'expression `[]` renvoie la liste vide ;
- l'expression `cons(X,L)` renvoie la liste dont le premier élément est l'entier `X` et la suite de la liste est la liste `L` ;
- l'opérateur `car(L)` renvoie le premier élément de la liste `L` ; si `L` est vide, le résultat est indéterminé ;
- l'opérateur `cdr(L)` renvoie la suite de la liste `L`, c'est-à-dire `L` privée de son premier élément ; si `L` est vide, le résultat est indéterminé ;
- on a aussi le test `vide(L)` qui renvoie VRAI si son argument est la liste vide, et FAUX sinon ; le résultat est indéterminé si `L` n'est pas une liste (c'est-à-dire si c'est un entier).

Exemple : la fonction suivante calcule le nombre d'élément de son argument s'il s'agit d'une liste (le résultat est indéterminé s'il ne s'agit pas d'une liste)

```
lgr(L) = si vide(L) alors 0 sinon ++ lgr(cdr(L))
```

3.1 n-ième élément

Écrire la fonction `ieme(N,L)` qui renvoie l'élément numéro `N` de la liste `L` (le premier a pour numéro... 1). Si `L` n'est pas une liste ou contient moins de `N` éléments, le résultat est indéterminé.

3.2 Concaténation

Écrire la fonction `concat(L,M)` qui renvoie la *concaténation* des deux listes `L` et `M`, c'est-à-dire leur mise bout à bout. Par exemple

```
concat([1,2,3],[4,5]) = [1,2,3,4,5]
```

3.3 Fusion

Écrire la fonction `fusion(L,M)` qui renvoie la *fusion* des deux listes **triées par ordre croissant** `L` et `M`, c'est-à-dire la liste **triée** contenant la réunion des éléments de `L` et de `M`. Par exemple

```
fusion([2,3,6],[1,4,5,7]) = [1,2,3,4,5,6,7]
```

3.4 Renversement

Écrire la fonction `renverser(L)` qui renvoie la liste dont les éléments sont ceux de `L`, **à l'envers**. Par exemple

```
renverser([1,2,3,4,5]) = [5,4,3,2,1]
```

NB : il y a au moins 4 algorithmes différents pour inverser une liste...